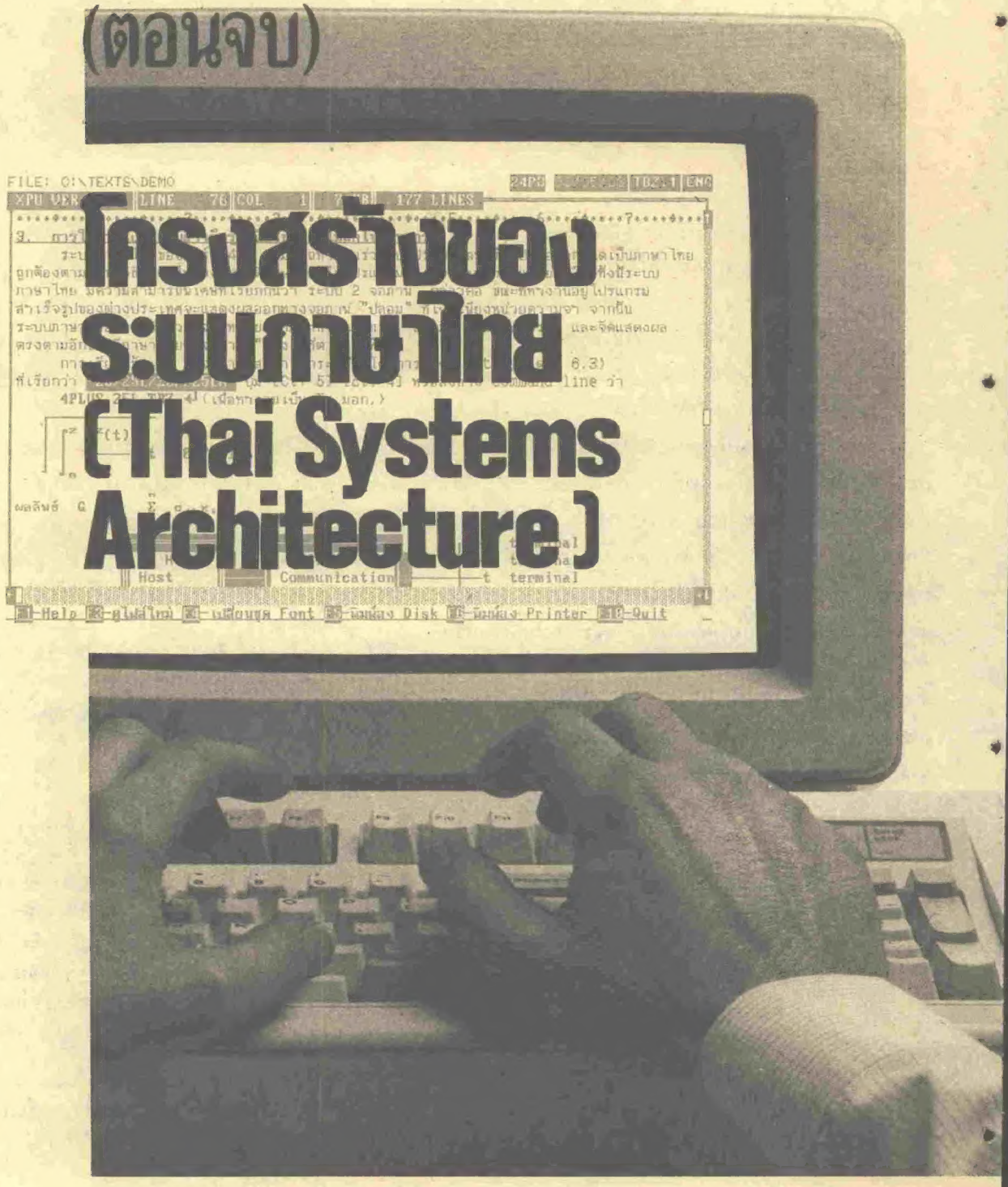
แนวทางออกแบบยูทิลิตี้รูทีน เพื่อการประมวลผลข้อมูลภาษาไทยด้วยความเร็วสูง (ตอนจบ)



โครงสร้างของระบบภาษาไทย (Thai Systems Architecture)

ทวีศักดิ์ กออนันตกูล

บทความตอนนี้จะเป็นตอนสุดท้ายที่ผู้เขียนจะให้ "แนวทาง" การออกแบบรูทีนต่าง ๆ โดยจะจัดรวบยอดให้เห็นภาพทั้งหมดของระบบภาษาไทยทั่ว ๆ ไปที่ใช้กันในประเทศไทย ภาพรวมที่เสนอให้เห็นนี้อาจเรียกได้ว่า เป็นสถาปัตยกรรมของระบบภาษาไทยเท่าที่มีผู้คิดสร้างขึ้นมาทั้งหมด ไม่ว่าท่านจะใช้ระบบภาษาไทยของใครอยู่ก็อาจจะถือได้ว่าระบบที่ท่านใช้อยู่เป็น subset ของภาพรวมที่เสนอในตอนนี้ แม้กระทั่งระบบยูนิกซ์ มินิ เมนเฟรม หรือแอปเปิ้ลแมคอินทอช โดยทำงานเป็นภาษาไทยก็อาจจะจัดอยู่ในโมเดลของ Thai Systems Architecture (TSA) นี้ได้เช่นเดียวกัน สิ่งที่ต่างกันก็อาจจะเพียงแค่ด้านของความครบถ้วนของการสร้างระบบ และการแยกส่วนของการทำงานว่าจะอยู่กับโฮสต์หรือเทอร์มินัลเท่านั้นเอง

ทำไมถึงต้องเรียกว่า "สถาปัตยกรรม"?

ผู้เขียนไม่ได้ตั้งใจจะเลือกคำหรูหราเพื่อนำมาใช้ตั้งชื่อระบบภาษาไทย แต่ในกรณีของ TSA นี้ หลังจากผู้อ่านได้ทราบถึงความตั้งใจและขอบเขตของ ที่ต้องการแล้วหวังว่าท่านผู้อ่านคงจะเข้าใจถึงความตั้งใจดีที่จะมอบ TSA นี้ให้เป็นแนวทางการออกแบบระบบภาษาไทยของทุกๆคน (ที่พอจะมองเห็นความดีของ TSA อยู่บ้าง) ก่อนที่จะอธิบายว่า TSA คืออะไร ใคร่ขอเน้นว่า TSA ไม่ใช่อะไรบ้าง

- TSA ไม่ใช่ระบบภาษาไทยหรือไทยการ์ดของผู้หนึ่งผู้ใด
- TSA ไม่ได้บังคับว่าวิธีการเรียกใช้ยูทิลิตี้รูทีนจะต้องทำในแนวหนึ่งแนวใดโดยเฉพาะ
- TSA ไม่ได้บังคับว่าจะต้องเป็น MS-DOS (จะเป็น UNIX, OS ของมินิ หรือเมนเฟรมอื่นๆ, หรือ OS/2 ก็ได้)

แต่ TSA คือ

- คำอธิบายถึงแนวทางการสร้างคอมพิวเตอร์ที่ทำงานกับภาษาไทยอย่างสมบูรณ์แบบโดยไม่อิงกับโครงสร้างทางฮาร์ดแวร์และโอเอสอันหนึ่งอันใด
- การรวบรวม data object ต่างๆ ที่จะต้องมีในระบบภาษาไทย
- การรวบรวมหัวข้องานโปรแกรมย่อย (procedure) ต่างๆ ที่จะต้องมีในระบบภาษาไทย ส่วนโปรแกรมย่อยเหล่านี้จะเขียนเป็นไลบรารี่ภาษาสูง หรือจะฝากไว้กับระบบภาษาไทยก็แล้วแต่จะทำ ขอให้สเปกการใช้งานตรงกัน ประโยชน์ย่อมมีมากมายมหาศาลแล้ว
- คำอธิบายถึงการไหลของข้อมูลต่างๆที่จะต้องเกี่ยวข้องกับภาษาไทย ทั้งในส่วนที่เกี่ยวกับ input/output device และส่วนที่เป็นการปฏิบัติงานประมวลผลของตัวข้อมูลในหน่วยความจำ

จากเป้าหมายที่กล่าวมาข้างบนนี้ เชื่อว่าคงจะไม่มีคำไหนที่จะอธิบายแนวการออกแบบนี้ได้ดีเท่ากับคำว่า "Thai Systems Architecture" (ในทำนองเดียวกันกับที่ไอบีเอ็มใช้คำว่า Systems Network Architecture(SNA), Systems Application Architecture(SAA) หรือ Information Systems Architecture(ISA) ทั้งนี้เพราะผู้เขียนถือว่าการสร้างระบบ (implementation) จริงๆนั้นทำได้หลายวิธีแม้ว่าจะใช้แนวทางการออกแบบอันเดียวกัน แต่อย่างไรก็ตามเพื่อให้ผู้อ่านมีโอกาสพิสูจน์ว่า TSA นั้นสามารถสร้างขึ้นได้จริงๆ และทำงานได้ดีในภาคปฏิบัติ ผู้เขียนขอแนะนำให้ทดลองกับระบบภาษาไทย 6FGPLUS รุ่น 2.00 ของเทลบิซ และ TSAB รุ่น 2.00 ของไทยซิสเต็มรีเสิร์ช (TSR) ทั้งสองระบบได้รับการออกแบบขึ้นมาตามแนว TSA เกือบทั้งหมด (แม้ว่ายังขาดรูทีนบางอันเพราะยังสร้างไม่เสร็จสมบูรณ์)

แรงจูงใจที่ทำให้ผู้เขียนเสนอแนวทางที่เป็นกลางและใช้ได้ทั่วไปนี้เกิดจากนโยบายของกระทรวงวิทยาศาสตร์ฯ ที่ประกาศว่าจะสนับสนุนการพัฒนาซอฟต์แวร์ที่ไม่ขึ้นอยู่กับฮาร์ดแวร์ (ซึ่งเป็นไปได้ยากในขณะนี้) แนวทางนี้จะเป็นไปได้จริงก็ต่อเมื่อผู้ผลิตฮาร์ดแวร์ทั้งหมดขายซอฟต์แวร์ระบบภาษาไทยภายใต้สถาปัตยกรรมเดียวกัน และคอมแพตทิเบิ้ลพอที่จะช่วยให้ซอฟต์แวร์ทำงานได้ดีในเครื่องที่ต่างกัน TSA ที่เสนอนี้เป็นเพียงแนวทางเพื่อก่อให้เกิดการพิจารณาการคิดสร้างสถาปัตยกรรมที่เป็นกลางขึ้นมาเพื่อประโยชน์ส่วนรวม

ฮาร์ดแวร์สำหรับ TSA

แนวความคิด TSA ไม่ได้ระบุว่าฮาร์ดแวร์ต้องเป็นอะไร ไม่ว่าจะเป็นพีซีโดดๆ หรือคอมพิวเตอร์แบบมีศูนย์กลางซึ่งจะเป็นแบบไหนก็ได้ ขอให้มีความสามารถในการรับข้อความภาษาไทยได้ (มักจะผ่านคีย์บอร์ด) แสดงผลทางจอภาพได้ (จะใช้ระบบอะไรก็ได้) และพิมพ์ตัวอักษรไทยหรือภาพบิตแมปกราฟิกออกมาได้ TSA ไม่ได้จำกัดรหัสของดิสเพลย์การ์ด (หรือของเทอร์มินัล) และรหัสของเครื่องพิมพ์แต่อย่างใด แม้ไม่มีสิทธิ์ดัดแปลงฮาร์ดแวร์เลย ความคิดแบบ TSA จะนำไปใช้กับการแสดงผลแบบกราฟิก และการพิมพ์ผลแบบกราฟิกออกทางเครื่องพิมพ์แบบจุดแมทริกซ์ก็ได้ หากจะพูดกันอย่าง

เอาจริงเอาจัง เราอาจจะระบุความต้องการขั้นต่ำบางอย่างเกี่ยวกับฮาร์ดแวร์ได้ 2 ประการคือ

1. ระบบ input/output เข้าสู่หน่วยความจำจะต้องเป็นแบบทีละ 8 บิต หรือแปลงให้เป็นแบบ 8 บิตในหน่วยความจำได้
2. ในการปฏิบัติงานจะยึดถือเอารหัส มอก. (มาตรฐานอุตสาหกรรม) แบบแอสกีเป็นตัวกลางสำหรับแลกเปลี่ยนเก็บข้อมูล และใช้เป็นรหัสภายในเครื่องเป็นหลัก ทั้งนี้ยกเว้นในกรณีที่มีความจำเป็นต้อง emulate หรือปลอมเป็นเครื่องรหัสอื่น เช่น มอก.EBCDIC หรือมหาวิทยาลัยเกษตรศาสตร์ ฯลฯ ก็จะใช้รหัสภายในเครื่องตามที่ระบุโดยจะเป็นรหัสใด ๆ ก็ได้

ระบบภาษาไทยต้อง "มี" อะไรบ้าง

ความสามารถในการประมวลผลข้อมูลภาษาไทยจะมีขึ้นโดยสมบูรณ์ได้นั้นจะต้องเกิดจากการออกแบบให้มีกระบวนการ (process) ต่างๆ ที่มีความรู้เกี่ยวกับอักขระวิธีภาษาไทย ในสมัยก่อนเราอาจจะถือเอาตัวกระบวนการเป็นสิ่งสำคัญที่สุด แต่ในสายตาของผู้เขียนพบว่า ถ้าจะวัดความสามารถของระบบภาษาไทยกันจริงๆแล้ว ควรจะถามว่ามีนั่นไหม มีนี่ไหม คำว่า "นั่น" และ "นี่" ในที่นี้ก็คือ data object ที่กำลังจะอธิบายในย่อหน้าต่อไปนั่นเอง เมื่อรู้ว่ามี data อะไรแล้ว จึงค่อยมาพูดถึงกระบวนการที่จะใช้ในการจัดการกับ data เหล่านี้

Data Object ใน TSA

data object คือ data (ชนิดหนึ่ง) ซึ่งเป็นที่ใช้เก็บรายละเอียดเกี่ยวกับการประมวลเป็นภาษาไทย โดยที่ระบบภาษาไทยระบบหนึ่งระบบใดอาจจะนำไปสร้างให้อยู่ในรูปข้อมูลแบบมีโครงสร้างลักษณะใดๆก็ได้ ตัวอย่าง data object ได้แก่

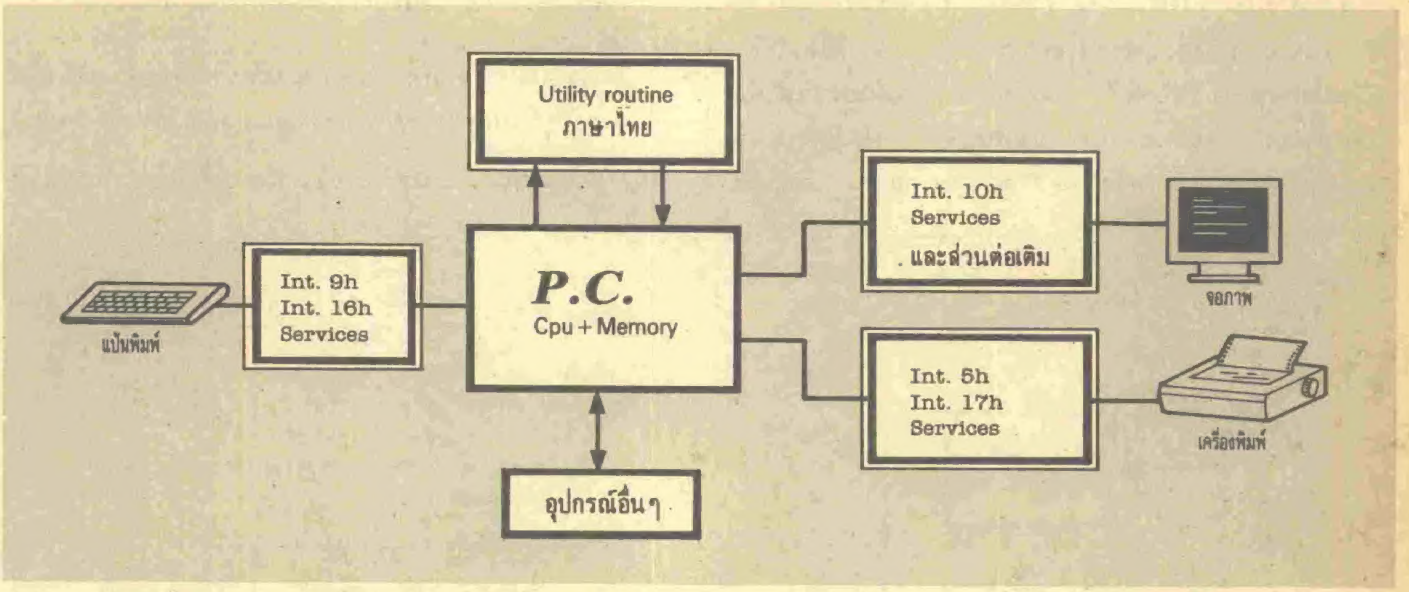
1. ซอฟต์แวร์สวิตช์ต่างๆ (boolean variables) คล้ายๆ ดิพสวิตช์ในฮาร์ดแวร์ สวิตช์เหล่านี้ใช้ควบคุมสภาพแวดล้อมการทำงานของเครื่อง สามารถเซ็ตให้ on(1) หรือ off(0) ได้จากโปรแกรมหรือคีย์บอร์ด ตัวอย่างสวิตช์ได้แก่
   - Thai Lock (บิตนี้หากมีค่า 1 ก็แปลว่าให้แป้นพิมพ์ทำงานเป็นไทย หากเป็น 0 หมายถึงการทำงานตามปกติเป็นภาษาอังกฤษ)
   - Composed Screen (1 = จัดระดับบนจอภาพ 0 = ไม่จัดระดับ)
   - Composed Printer (1 = จัดระดับบนเครื่องพิมพ์ 0 = ไม่จัดระดับ)
   - Keyboard Check (1 = ตรวจสอบความถูกต้องของการป้อนอักขระไทย 0 = ไม่ตรวจสอบ)
   - (ฯลฯ)

ระบบไหนมีสวิตช์มากก็หมายความว่าควบคุมการทำงานได้ละเอียดดี (แต่อย่าให้มีพร่ำเพรื่อ) ระบบไหนอนุญาตให้โยกสวิตช์ได้จากโปรแกรมหรือบางครั้งโดยใช้ Hot key ก็ถือได้ว่าช่วยให้เกิดความสะดวก ในทางตรงกันข้ามหากมีสวิตช์มาก แต่ออกแบบไว้ไม่ดีหรือขาดระบบบริหารที่ดีก็อาจจะเกิดความสับสนแก่ผู้ใช้

2. ตารางต่างๆ (tables) ซึ่งเป็นกลไกสำคัญที่ช่วยให้กระบวนการ (process) การประมวลผลข้อมูลภาษาไทยทำได้ง่ายและรวดเร็ว ตัวอย่างตารางได้แก่
   - Char Type Table (ตารางบอกคุณสมบัติด้านต่างๆ ของตัวอักษรที่เราอยากทราบ)
   - I/O Conversion Tables (ตารางแปลงรหัสไปมาระหว่างรหัสภายในคอมพิวเตอร์กับรหัสของอุปกรณ์อินพุตเอาต์พุตแต่ละตัว อาทิ Keyboard table, CRT table, Printer Table)
   - Character Generator Tables (ตารางแปลงรหัสตัวอักษรให้กลายเป็นภาพตัวอักษรแบบบิตแมปเพื่อใช้กับระบบที่แสดงภาพแบบกราฟิก หรือกำหนดตัวพิมพ์ออกทางเครื่องพิมพ์โดยวิธีดาวน์โหลด)
   - ASCII Conversion Tables (ตารางแปลงรหัสจากรหัสมาตรฐานเป็นรหัสที่ต้องการใช้งานจริง ๆ ซึ่งอาจจะไม่ใช่รหัสมาตรฐาน)
   - Control Tables (เช่น ตารางควบคุมการเว้นบรรทัดการพิมพ์ภาษาไทย ตารางควบคุมอุปกรณ์ CRT controller ตารางกำหนดแอดเดรสของ VRAM โหมดต่างๆ หรือแอดเดรสของบรรทัดต่างๆ บนจอภาพ หรือแม้กระทั่งตารางสำหรับควบคุมสวิตช์ต่างๆ ที่กล่าวมาแล้วข้างต้น)

มี data แล้วจึงจะมีโปรแกรม

หลังจากที่ผู้สร้างระบบภาษาไทยได้คำนึงถึงdata object ต่างๆ เสร็จแล้วก็จะต้องนำ data เหล่านั้นมาใช้งานควบคู่กับระบบอินพุตเอาต์พุตที่กำลังศึกษาอยู่ เพื่อสร้างเป็นหัวข้องานที่จะต้องเขียนเป็นโปรซีเจอร์หรือฟังก์ชันตามลักษณะการใช้งานที่จะเกิดขึ้นจริง นั่นคือการไปดักการทำงานเดิมของระบบที่มาจากต่างประเทศ (ซึ่งทำงานกับข้อมูลภาษาอังกฤษเพียง



รูปที่ 1 การจัดทำระบบภาษาไทยระดับไบออสของไอบีเอ็มพีซี ส่วนที่เป็นกรอบคู่ หมายถึงซอฟต์แวร์ที่จะต้องมีความรู้ภาษาไทย

อย่างเดียว) ให้โปรแกรมสามารถทำงานแสดงผลเป็นไทยได้อย่างถูกต้องตามข้อกำหนดรหัสอักขระที่ต้องการ ยกตัวอย่าง เช่น ในกรณีของคอมพิวเตอร์ที่ใช้ PC-DOS หรือ MS-DOS เป็นโอเอส ผู้สร้างระบบภาษาไทยก็จะต้องเขียนโปรแกรมส่วนต่อไปนี้เสียใหม่ (ดูรูปที่ 1)

- Interrupt 9h เซอร์วิสรูทีน - เพื่อให้สามารถสลับแป้นพิมพ์ไทย/อังกฤษได้อย่างถูกต้อง โดยแป้นพิมพ์ภาษาอังกฤษมีความหมายคงเดิม ส่วนแป้นพิมพ์ไทยมีความหมายตรงตามมาตรฐานอุตสาหกรรม
- Interrupt 10h เซอร์วิสรูทีน - เพื่อให้รู้จักกับการเขียนจอภาพเป็นภาษาไทยแต่มีอินเตอร์เฟสกับแอพพลิเคชั่นตามแนวของไอบีเอ็มพีซีไบออส รายละเอียดของโปรแกรมส่วนนี้ขึ้นกับไทยการ์ดที่ใช้ (ดูโครงสร้างของไทยการ์ดแบบต่างๆ ได้ใน "จอคอมพิวเตอร์ไทยแบบต่างๆ" จากไมโคร ฉบับที่ 41)
- Interrupt 17h เซอร์วิสรูทีน - เพื่อให้สามารถพิมพ์ข้อความออกมาเป็นภาษาไทยได้อย่างถูกต้องตามอักขระวิธี รายละเอียดส่วนนี้ขึ้นอยู่กับรหัสของเครื่องพิมพ์ที่ใช้ ซึ่งบางรายอาจใช้วิธีดาวน์โหลด "ภาพตัวอักษร" (CG table) จากโปรแกรมเข้าสู่เครื่องพิมพ์
- Interrupt 5h เซอร์วิสรูทีน - เพื่อให้สามารถพิมพ์จอภาพออกทางเครื่องพิมพ์อย่างถูกต้อง

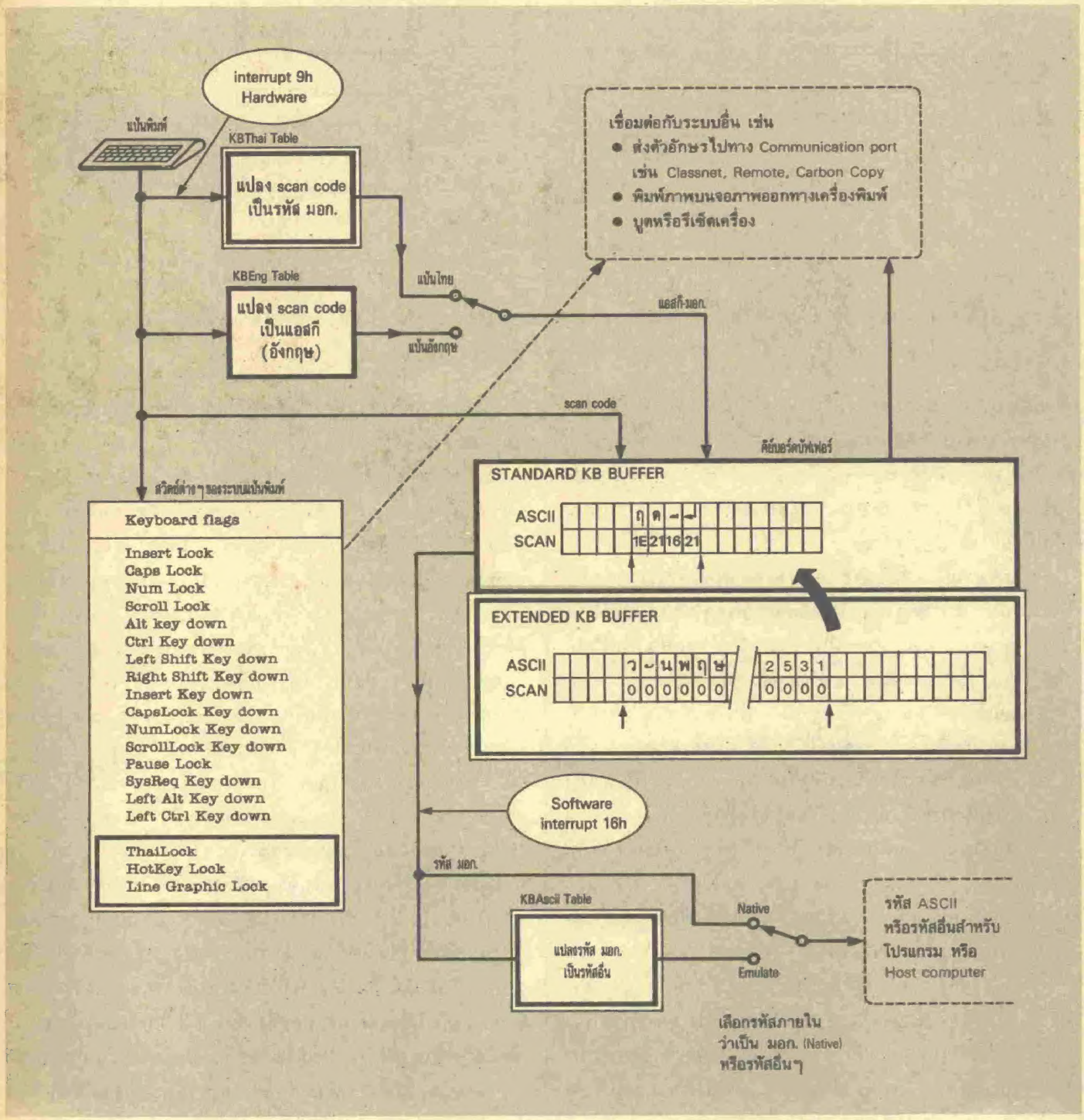
นอกจากนี้ก็จะต้องมีโปรแกรมย่อยๆอีกจำนวนมาก เพื่อช่วยในการเปลี่ยนค่าของสวิตช์ต่างๆ ระหว่างการทำงานกับเปลี่ยนค่าของ table ต่างๆด้วย โปรแกรมย่อยๆเหล่านี้หากมิได้วางแผนการทำงานไว้ หรือสร้างโปรแกรมขึ้นมาแบบค่อยเป็นค่อยไป พบอะไรก็เติมๆลงไป อาจจะเกิดปัญหาในการทำงานได้เหมือนกัน สวิตช์หลายๆตัวมีความสัมพันธ์กันอย่างใกล้ชิด เช่น สวิตช์ที่ใช้ในการจัด/ไม่จัดจอภาพ (Composed Screen) หากอยู่ในตำแหน่งที่ "ไม่จัด" การปฏิบัติงานของ keyboard check (ตรวจสอบความถูกต้องของลำดับการป้อนอักขระไทย) แม้จะอยู่ในตำแหน่ง "เช็ค" ความถูกต้องก็ควรจะถูกสั่งให้ลืมไปก็ได้ เพราะการแสดงภาษาไทยแบบไม่จัดระดับน่าจะปล่อยให้ผู้ใช้เครื่องป้อนตัวอักษรใด ๆ ก็ได้ บางครั้งตัวอักษรเหล่านั้นเป็น line graphic character แต่รหัสตรงกับสระอูหรือสระอี ก็น่าจะปล่อยให้อยู่ติดกันได้

โครงสร้างของ TSA ในที่นี้จะเอ่ยถึงเฉพาะตัว data object และหัวข้อ procedure ต่างๆที่จะต้องมีโดยสังเขป ไม่ได้กังวลว่าแต่ละโปรซีเจอร์จะเรียกใช้กันอย่างไร หากจะทำให้ได้ถึงขั้นมาตรฐานกันจริงๆแล้ว โปรซีเจอร์เดียวกันของทุกบริษัท (ที่เป็น TSA ด้วยกัน) จะต้องมีอินเตอร์เฟสเดียวกันหมด เพื่อว่าซอฟต์แวร์เฮ้าส์อิสระจะสามารถพัฒนาระบบงานดีๆขึ้นมาได้ โดยใช้งานได้ดีกับระบบภาษาไทยตามมาตรฐานเดียวกันได้ทุกระบบ ทั้งๆที่ฮาร์ดแวร์ต่างกัน คณะกรรมการพัฒนาวิทยาศาสตร์และเทคโนโลยีก็อาจจะช่วยให้มีการตกลงกันว่าจะมีอะไรบ้าง (data object และ procedure) แบบไหนที่พอจะเป็นมาตรฐานกลางได้ เมื่อได้แล้วผู้ขายระบบภาษาไทยจึงจะได้นำไปสร้างหรือดัดแปลงระบบเดิมของตนให้มี application program interface (หรือ API ตามภาษาของไอบีเอ็ม) ตรงตามที่กำหนดไว้

โครงสร้างของระบบแป้นพิมพ์

รูปที่ 2 แสดงถึงโครงสร้างของระบบแป้นพิมพ์ที่ใกล้เคียงกับไบออสของไอบีเอ็มพีซี แต่มีการเพิ่มเติมชิ้นส่วนสำคัญเข้าไปบางอย่างเพื่อให้ทำงานภาษาไทยได้ดี หากมีครบถ้วนตามนี้ก็น่าจะเรียกได้ว่าตรงตาม Thai Systems Architecture ส่วนที่เพิ่มเติมจากระบบภาษาอังกฤษที่สำคัญๆ ได้แก่

- ตารางแปลงรหัสจากสแกนโค้ดของคีย์บอร์ดให้เป็นรหัส มอก.620-2529 (หรือที่เรียกกันว่า รหัส สมอ.)
- Extended keyboard buffer เพื่อใช้ต่อเพิ่มขนาด



รูปที่ 2 โครงสร้างของระบบแป้นพิมพ์ภาษาไทย มองจากการไหลของข้อมูล ส่วนที่ตีกรอบคู่หมายถึง data object ที่สร้างขึ้นเพิ่มเติมจากของไอบีเอ็ม เพื่อให้ทำงานเป็นภาษาไทย

ของบัฟเฟอร์เดิมซึ่งมีขนาดเพียง 16 ตัวอักษร (บัฟ–เฟอร์เดิมไม่ควรย้ายไปอยู่ที่อื่น) ให้ยาวขึ้นเท่าที่ต้องการ

- KB Ascii Table เพื่อใช้แปลงรหัสจาก มอก. เป็นรหัสภายในเครื่องที่ต้องการ (ไม่จำเป็นต้องใช้รหัสมาตรฐานเสมอไป)

ในการปฏิบัติงานกับระบบแป้นพิมพ์นี้ ผู้ออกแบบระบบภาษาไทยจะต้องมีสวิตช์ต่างๆ ใช้งานควบคู่ไปกับตารางแปลงรหัสที่เพิ่มเข้าไปด้วย สวิตช์ต่างๆที่เกี่ยวข้องกับแป้นพิมพ์ที่จำเป็นผู้เขียนได้สรุปไว้ในรูปที่ 3

หลังจากเห็นโครงสร้างทางข้อมูลของระบบแป้นพิมพ์แล้วผู้ออกแบบระบบภาษาไทยคงจะต้องคำนึงถึงวิธีการต่างๆ

ซอฟต์แวร์สวิทช์	ความหมาย
Thai Lock	เป็นผลจากการกดปุ่มบนแป้นเพื่อสลับความหมายของแป้นพิมพ์จากแป้นไทย เป็นแป้นอังกฤษ แต่ละระบบในปัจจุบันยังใช้ปุ่มต่างกัน เช่น ปุ่ม Alt ซ้ายมือ, ปุ่ม Scroll Lock, ปุ่ม Ctrl+Shift หากสวิทช์ ThaiLock เป็น 1 หมายความว่าขณะนี้แป้นพิมพ์เป็นไทย หากเป็น 0 แปลว่าแป้นพิมพ์เป็นภาษาอังกฤษ
HotKey Lock	เป็นผลจากการกดปุ่ม เพื่อเริ่มต้นป้อน hotkey เมื่อ hotkey ได้รับการป้อนจนจบแล้ว ค่าของสวิทช์ Hotkey Lock จะกลับไปมีค่าเป็น 0 ปุ่มของ hotkey จะเป็นปุ่มใดๆก็ได้ ตามแต่ผู้ออกแบบจะกำหนด เช่น ปุ่ม Ctrl+ NumericPad 5
LineGraphic Lock	เป็นผลจากการกดปุ่มเพื่อเริ่มต้นป้อนตัวอักษรแบบ line graphic (รหัสของ line-graphic character จะตรงกับรหัสของอักษรไทย) ในระบบภาษาไทย 6FGPLUS รุ่น 2.00 อนุญาตให้ป้อนตัวอักขระแบบ linegraphic ได้ตลอดเวลากับทุกโปรแกรม โดยการกดปุ่ม CapsLock ค้างไว้เสมือนการ shift ตามด้วยปุ่มต่างๆบนแผง numeric pad
Native / Emulate	เป็นผลจากการสั่งให้ระบบคอมพิวเตอร์รหัสมาตรฐาน (มอก.) ทำงานโดยใช้รหัสภายในที่แตกต่างไปจากรหัส มอก. ทั้งนี้เพื่อใช้งานกับโปรแกรมหรือฐานข้อมูลซึ่งไม่ได้เขียนขึ้นสำหรับรหัส มอก. โดยตรง หากซอฟต์แวร์สวิทช์ตัวนี้ถูกโยกมาทาง emulate ตัวอักษรใน KB buffer (รูปที่ 1) ซึ่งเป็น มอก. จะถูกแปลงเป็นรหัสอื่นที่ต้องการ ก่อนจะส่งค่าตัวอักษรนั้นไปยังแอปพลิเคชัน หรือคอมพิวเตอร์กลาง

รูปที่ 3 ตัวอย่างความหมายของสวิตช์ต่างๆ ที่มีใช้กันในระบบแป้นพิมพ์ภาษาไทย

ที่จะจัดการกับข้อมูลเหล่านี้ ซึ่งในส่วนของแป้นพิมพ์นั้นผู้เขียนขอยกตัวอย่างดังนี้

ก. การแจกแจง keyboard scan code เพื่อแยกงานส่วนที่เป็นการส่งตัวอักษรเข้าบัฟเฟอร์ หรือเป็นการเปลี่ยนค่าของสวิตช์ตัวหนึ่งตัวใดหรือหลายตัวตามปุ่มที่กดเข้ามา ตัวอย่างโปรแกรมส่วนนี้สามารถอ่านได้จาก BIOS listing ของไอบีเอ็มพีซี (ในคู่มือ Technical Reference Manual) หากอ่านจนเข้าใจแล้วก็สามารถเขียนโปรแกรมที่ทำงานเพิ่มเติมเกี่ยวกับการกำหนดค่าของซอฟต์แวร์สวิตช์ต่างๆ ตามรูปที่ 3

ข. การกำหนดค่าของตารางต่างๆ ซึ่งแม้ว่าจะไม่ใช่สิ่งจำเป็นขั้นพื้นฐาน แต่หากมีก็จะสามารถทำให้ผู้ใช้สามารถเลือกได้ว่าจะจัดแป้นพิมพ์สไตล์ไหน (เกษมณี หรือปัตตะโชติ หรือตามมาตรฐานอุตสาหกรรม) ช่วยให้ผู้ใช้สามารถใช้งานแอพพลิเคชั่นโปรแกรมเก่าๆ ที่เขียนผูกไว้กับรหัสอักขระไทยแบบอื่นๆ เช่น เกษตร หรือ EBCDIC ได้โดยไม่ต้องแก้ไข

ค. การปฏิบัติงานขั้นสูง ตัวอย่างเช่น การเพิ่มขนาดของคีย์บอร์ดบัฟเฟอร์จากมาตรฐานเดิมของพีซี (16 ตัวอักษร) ให้มากขึ้นหลายๆเท่าได้ นอกจากจะทำให้พิมพ์ล่วงหน้าได้มากขึ้นแล้วยังสามารถทำงานอื่นๆ เช่น keyboard macro หรือป้อนชื่อ วันที่ เวลาได้อย่างรวดเร็ว นอกจากนี้บัฟเฟอร์ของคีย์บอร์ดยังสามารถใช้งานเป็น Interprocess communication ได้ด้วย ตัวอย่างเช่น ขณะอยู่ในดีเบส เราอาจจะเรียกใช้โปรแกรมๆหนึ่ง เพื่อทำงานบางอย่างที่ดีเบสๆไม่ถนัด เมื่อโปรแกรมนั้นเสร็จงาน อาจคืนคำตอบไว้ให้ในคีย์บอร์ดบัฟเฟอร์ได้ และดีเบสจะอ่านคำตอบได้ทันทีโดยใช้คำสั่งที่เหมือนกับการรับข้อความจากมนุษย์ เป็นต้น ในกรณีของแมคอินทอช หรือพรีเซนเทชัน แมเนจเจอร์ของ OS/2 คำว่าคีย์บอร์ดบัฟเฟอร์ จะกลายเป็น event buffer ซึ่งจะเก็บทั้งตัวอักษรและตำแหน่งของเม้าส์ที่เรากดลงไป

โครงสร้างของระบบแสดงภาพ

ในสมัยแรกที่มีการประดิษฐ์โปรแกรมประเภทไทยไดร์เวอร์ขึ้นมา เราอาศัยวิธีการเขียนรูทีนประเภท Console Output ขึ้นมาแทนที่ของต่างประเทศ วิธีนี้ใช้ได้ผลดีกับระบบ CP/M และ CP/M-86 เพราะระบบนี้มีมาตรฐานการแสดงผลทางจอภาพเพียงแบบเดียว คือถือว่าการติดต่อกับยูสเซอร์นั้นผ่านอุปกรณ์ที่เรียกว่าเทเลไทป์ หรือเครื่องพิมพ์แบบมีคีย์บอร์ดนั่นเอง มาในช่วงประมาณปี 1972 เป็นต้นมา การใช้จอเทอร์มินัล

เริ่มจะมากขึ้นเพราะเร็วกว่าเทเลไทป์มาก แต่จอภาพในสมัยนั้นก็ทำงานได้แค่ทดแทนเครื่องเทเลไทป์เท่านั้นเอง ดังนั้นระบบการแสดงผลทางจอภาพของคอมพิวเตอร์ก่อนปี 1972 จึงเป็นเพียงประเภท character I/O คือ คอมพิวเตอร์รับ/ส่งข้อความแบบทีละตัวอักษร ไม่ว่าจะเป็นเมนเฟรม มินิ หรือไมโครคอมพิวเตอร์

จอภาพรุ่นต่อๆมาได้มีผู้เพิ่มความสามารถพิเศษเข้าไปบางอย่าง เช่น การกำหนดตำแหน่งของเคอร์เซอร์ หรือการกำหนดแอตทริบิวต์ของตัวอักษร สามารถทำได้โดยการประสมคำสั่งเหล่านี้ปนไปกับข้อความที่ส่งมายังจอภาพได้ การกำหนดตำแหน่งสามารถกำหนดได้ทั้งแบบตำแหน่งสัมบูรณ์ (บอกตำแหน่งโรว์และคอลัมน์) และการย้ายเคอร์เซอร์ไปทางซ้าย/ขวา บน/ล่าง ด้วยวิธีการนี้เองระบบเทเลไทป์จึงสามารถทำเป็นภาษาไทยแบบ 3 ระดับ (25 แถว กลายเป็น 8 บรรทัดภาษาไทย) ได้ง่าย เพราะโปรซีเจอร์จัดระดับภาษาไทยทำหน้าที่ "ขยาย" สตริงให้กลายเป็นสตริงประสมกับคำสั่งให้เลื่อนเคอร์เซอร์ขึ้นลง ตัวอย่างเช่น โปรแกรมส่งสตริง "ชื่นใจ" มาให้ระบบแสดงภาพ ระบบแสดงภาพก็จะแปลง "ชื่นใจ" ให้กลายเป็น "ช/ซ้าย/ขึ้น/ ◌ี/ลง/ขวา/ซ้าย/ขึ้น/ ◌ื/ลง/ขวา/น/ใ/จ" ส่งไปยังจอภาพ หรือส่งให้โปรแกรมแสดงภาพคำว่า ซ้าย, ขึ้น, ขวา, ลง ข้างบนนี้หมายถึงคำสั่งประเภท escape sequence แต่ละคำสั่งมีความยาว 2-3 ไบต์

ยุคของเทเลไทป์สิ้นสุดลงไปพร้อมๆกับการมีเทอร์มินัลแบบฉลาด (intelligent terminal) และพีซี เพราะการทำงานกับเทอร์มินัลสามารถทำอะไรเพิ่มขึ้นได้อีกมากมาย ดังจะเห็นได้จากฟังก์ชันต่างๆ สำหรับจอภาพของไอบีเอ็มพีซีซึ่งมีอยู่ 16 ฟังก์ชัน เก็บไว้ในไบออส ภายหลัง (ปี 1987) มีการขยายเพิ่มขึ้นอีกหลายฟังก์ชัน โปรแกรมที่มีชื่อเสียงหลายโปรแกรมนิยมเรียกใช้วิดีโอฟังก์ชันของไบออสโดยตรงแทนที่จะเรียกผ่าน Console ของดอส (หรือ CON:) ซึ่งเป็นความคิดแบบเทเลไทป์ ผลที่เกิดขึ้นก็คือการทำภาษาไทยบนไอบีเอ็มพีซีสลับซับซ้อนกว่าของเทเลไทป์มาก เพราะจะต้องปรับการทำงานของวิดีโอฟังก์ชันทั้ง 16 อันให้มีความสามารถทำงานเป็นภาษาไทยได้ด้วย รูปที่ 4 แสดงให้เห็นถึงการไหลของข้อมูลเมื่อมีการเรียกใช้วิดีโอฟังก์ชันเพื่อการแสดงตัวอักษร 1 ตัวบนจอภาพ

อันที่จริงระบบการแสดงภาพในรูปที่ 4 มีส่วนที่เรียกว่า Shadow screen processing ขึ้นมาอีกส่วนหนึ่ง ซึ่งไม่จำเป็นสำหรับการใช้งานตามปกติ แต่เป็นกลไกพิเศษซึ่งผู้เขียนออกแบบขึ้นสำหรับการทำงานร่วมกับซอฟต์แวร์ต่างประเทศโดยไม่มีการแก้ไข และสำหรับการทำงานกับข้อมูลและโปรแกรมที่ใช้รหัสอื่นใดก็ได้ที่ไม่ใช่ มอก. ดังจะได้อธิบายต่อไป

การจัดระดับของตัวอักขระบนจอภาพแบบปกติ

ส่วนนี้จะตรงกับวิดีโอฟังก์ชันหมายเลข 0AH(write character), 09H(write attribute/character) และ 0EH (write teletype) การวางตัวอักษรจะวางครั้งละหนึ่งตัว หรือหลายๆตัวแต่ซ้ำกัน เมื่อนำมาดัดแปลงเป็นไทย ผู้สร้างระบบภาษาไทยก็จะต้องนำตัวอักษรมาพิจารณาว่าจะต้องวางตรงไหนโดยเทียบกับตำแหน่งเคอร์เซอร์ปัจจุบัน ผลที่ได้มี 3 กรณีคือ

ก. วางตรงตำแหน่งเคอร์เซอร์ (และเลื่อนเคอร์เซอร์ไปทางขวา) กรณีนี้ใช้กับตัวอักษรอังกฤษและพยัญชนะไทย

ข. วางตรงทางซ้ายของเคอร์เซอร์ (เคอร์เซอร์หยุดนิ่ง) เหนือตัวอักษรที่อยู่ทางซ้ายมือเดิม กรณีนี้ใช้กับสระหรือวรรณยุกต์ที่ต้องวางเหนือระดับบรรทัด

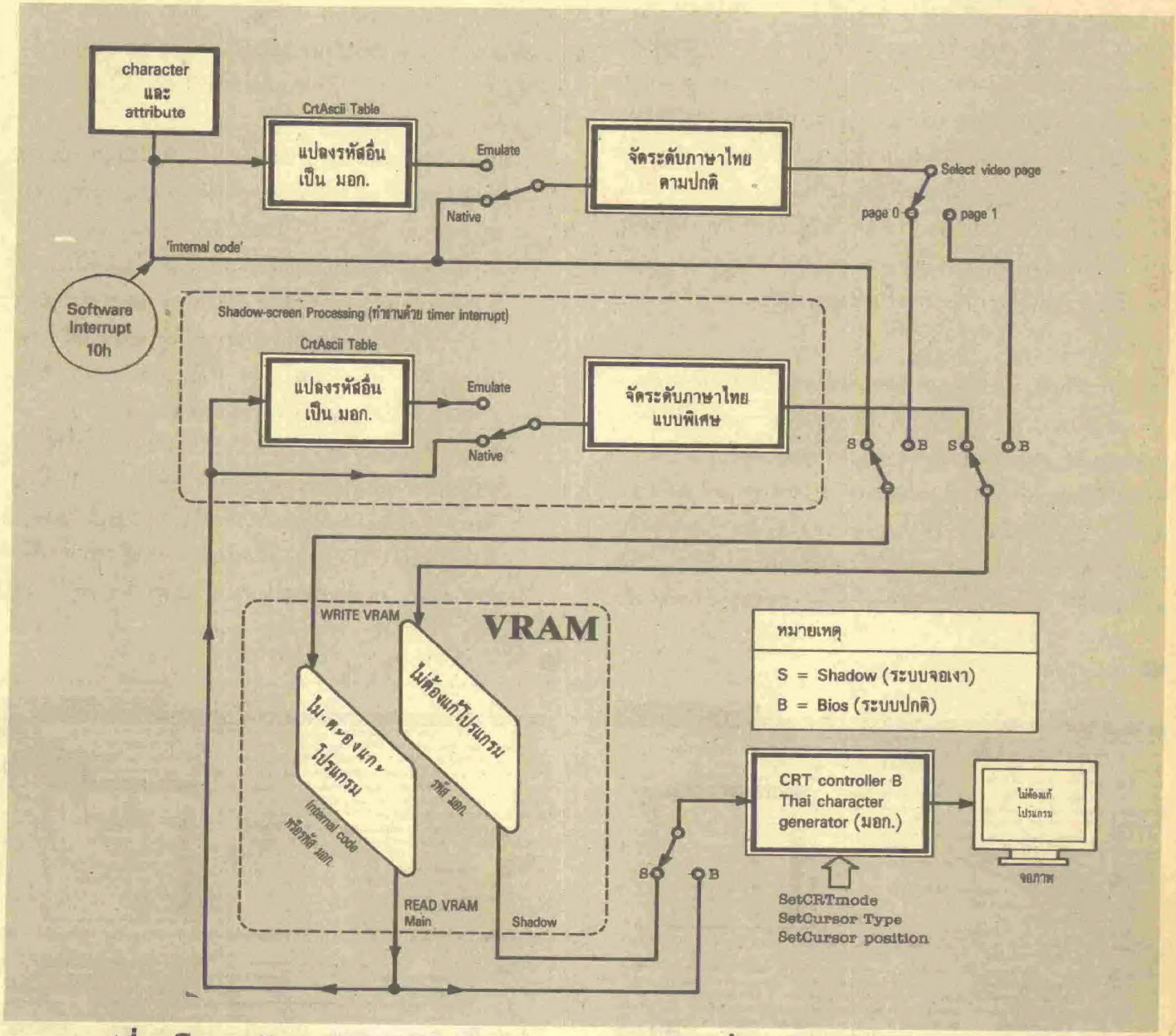
ค. วางตรงทางซ้ายของเคอร์เซอร์ (เคอร์เซอร์หยุดนิ่ง) ใต้ตัวอักษรที่อยู่ทางซ้ายมือเดิม กรณีนี้ใช้กับ ◌ุ ◌ู (พินทุ)

การจัดระดับของตัวอักขระบนจอ Shadow Screen

Shadow Screen หรือ "จอเงา" เป็นกลไกการแสดงภาพสมัยใหม่ที่มีผู้ทำขึ้นเพื่อต่อสู้กับปัญหาที่ว่า โปรแกรมบางโปรแกรมดื้อมากเขียนผลลัพธ์ออกทาง VRAM โดยตรง ทำให้ระบบการจัดระดับของอักขระตามปกติไม่ได้รับทราบ จึงไม่ได้จัดรูปการแสดงภาพเสียใหม่ให้ตรงตามอักขระวิธี ระบบ "จอเงา" นี้ทางต่างประเทศเป็นผู้คิดขึ้นก่อน เพื่อใช้ในการ "ขัง" ภาพจากจอภาพของโปรแกรมประเภทดื้อดึงให้ปรากฏอยู่ในวินโดว์ซึ่งเป็นส่วนหนึ่งของจอเงา ระบบนี้ใช้ใน DesqView (จอเงาเป็น text) และ Microsoft Windows (จอเงาเป็นกราฟิก) ผู้เขียนได้ทดลองนำมาใช้กับระบบภาษาไทย พบว่าวิธีการนี้มีประโยชน์มากและช่วยให้ไทยการ์ดที่ใช้รหัสภาษาไทยตามมาตรฐาน มอก.ทำงานกับโปรแกรมรหัสอื่นได้ดีและโดยไม่มีข้อจำกัดใดๆ ไม่ว่าจะเป็นแบบ 8 บรรทัดหรือ 25 บรรทัด นอกจากนี้ยังสามารถทำให้ใช้งานโปรแกรมต่างประเทศกับข้อมูลภาษาไทยแบบจัดระดับได้โดยไม่ต้องแก้ไข

หลักการทำงานของระบบ Shadow screen ก็คือการยอมปล่อยให้แอพพลิเคชั่นโปรแกรมแปลกๆ ที่เขียนลง VRAM โดยตรงหรือเขียนผ่านไบออส interrupt 10h ทำงานกับหน่วย

ความจำของ "จอจริง" (main screen) ไปตามปกติ แต่ฮาร์ดแวร์ของระบบถูกสวิตช์ไปให้จอภาพแสดงภาพของ VRAM ส่วนที่เป็น "จอเงา" ซึ่งเป็นคนละส่วนกับ "จอจริง" ที่แอพพลิเคชั่นใช้งานอยู่นั้นเราจะไม่เห็น เห็นแต่จอเงา ระบบที่ทำงานแบบ shadow screen จะมีโปรแกรมพิเศษฝากไว้กับ timer interrupt ซึ่งจะถูกเรียกใช้งานวินาทีละ 18.2 ครั้งโดยอัตโนมัติ โปรแกรมพิเศษส่วนนี้จะทำหน้าที่คัดไทย จากจอจริงมาสู่จอเงา หน่วยความจำส่วนที่เป็น "จอจริง" เสมือนเป็นภาพหน้าจอแบบไม่จัดระดับ และอาจจะเป็นรหัสอื่นที่ไม่ใช่ มอก. ก็ได้ การคัดไทย (ดูรูปที่ 4) โดย Shadow screen processing จะทำหน้าที่แปลงรหัสให้เป็น มอก. (หากจำเป็นต้องทำ) แล้วจึงจะนำไปเขียนลงหน่วยความจำที่เป็น "จอเงา" ตามโครงสร้าง



รูปที่ 4 โครงสร้างของระบบจอภาพภาษาไทย มองจากการไหลของข้อมูล สภาพการทำงานปกติคือ Native และ B ในภาพแสดงการทำงานแบบ Shadow screen รหัส มอก. สวิตช์ต่างๆ จึงอยู่ในตำแหน่ง Native (มอก.) และ S ในกรณีที่ต้องการทำงานกับข้อมูลรหัสอื่นๆ จะต้องโยกสวิตช์จาก Native ไปยัง Emulate เพื่อให้มีการแปลงรหัสก่อนการจัดระดับ ระบบที่แสดงนี้ถือว่า Character generator มีตัวอักษรตาม มอก.เพียงรหัสเดียว

การแสดงผลของการ์ดอย่างถูกต้อง ผลก็คือ ผู้ใช้เห็นภาพจากจอเงาแบบจัดระดับอย่างถูกต้อง ดังแสดงในรูปที่ 5 (ภาพ ข.)

สิ่งที่ผู้พัฒนาระบบภาษาไทยจะต้องพิจารณากันให้ดีก็คือ สูตรของการคัดไทย ว่าจะทำกันอย่างไรดี จึงจะรู้สึกว่าเป็นธรรมชาติ โดยไม่ทำให้เครื่องช้าลง บางระบบก็อาศัยวิธีตรวจสอบดูว่า "จอจริง" มีอะไรเปลี่ยนไป ถ้าเปลี่ยนจึงจะทำการคัดไทย หากไม่เปลี่ยนก็จะข้ามไป บางระบบใช้ฮาร์ดแวร์อินเตอร์รัพต์ ติดเข้าไปกับการเขียนข้อมูลสู่หน่วยความจำใน "จอจริง" วิธีนี้ช่วยให้ทราบทันทีว่า "จอจริง" มีอะไรเปลี่ยนแปลงบ้าง และเปลี่ยนตรงไหน บางระบบใช้สูตรการคัดไทยด้วยความถี่ต่ำ ๆ เพื่อไม่ให้เครื่องช้าลง ทั้งนี้ยกเว้นบรรทัดที่มีเคอร์เซอร์ ซึ่งจะต้องมีการคัดไทยอยู่ค่อนข้างถี่ ระบบหลังนี้ใช้ในระบบ 4FGPLUS และ TSAB ซึ่งผู้เขียนเป็นผู้ออกแบบ พบว่าให้ความรู้สึกที่ดีมาก แม้ขณะที่กำลังป้อนข้อความภาษาไทยด้วยความเร็วสูงก็จะสามารถแสดงตัวอักษรทุกตัวได้ทันเวลา

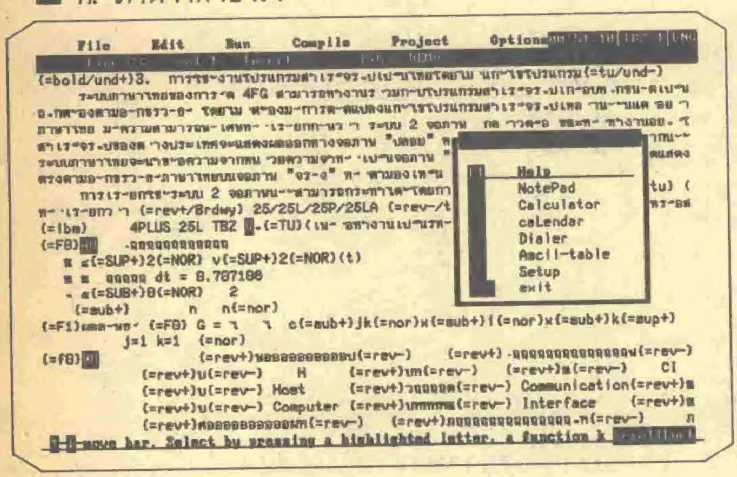
การวัดความเร็วของซีพียูขณะทำงานในระบบ Shadow screen นั้นสามารถวัดได้โดยการพิมพ์ข้อความจากไฟล์ยาว ๆ สู่จอภาพแล้วจับเวลา แต่ทั้งนี้จะต้องตรวจสอบดูด้วยว่าข้อความทุกบรรทัดปรากฏบนจอครบถ้วนไม่มีส่วนหนึ่งส่วนใดขาดหายไป อีกวิธีหนึ่งก็คือใช้ editor ของไซด์คิกหรือเทอร์โบ-ปาสคาลทำงานกับไฟล์ภาษาไทย ทดสอบความรู้สึก "เป็นธรรมชาติ" โดยการกด PgUp หรือ PgDn ค่อนข้างถี่ ๆ ระบบที่ดีจะต่างจากระบบไม่ดีอย่างเห็นได้ชัด อนึ่ง การวัดความเร็วด้วย Norton SI หรือ Landmark SPEED อาจให้คำตอบที่ขัดกับความจริงได้ การจับเวลาการ type file จะให้ผลชัดเจนกว่า

จะใช้ Shadow screen ดีไหม?

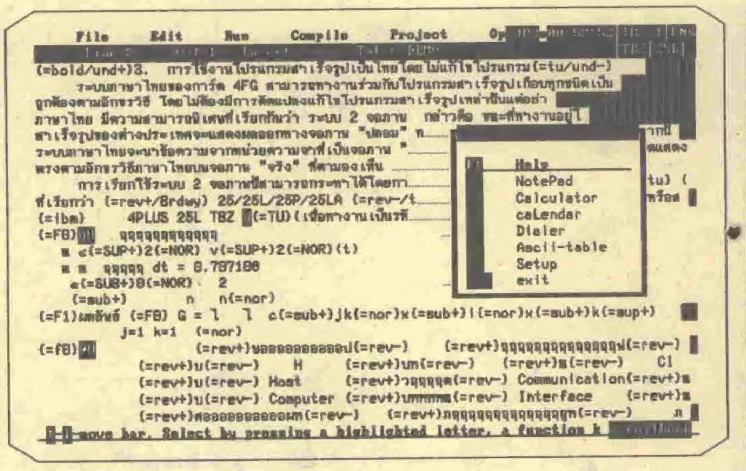
วิธีการแบบ Shadow screen เมื่อสร้างสำเร็จใหม่ ๆ ผู้เขียนค่อนข้างจะตื่นเต้นกับความมหัศจรรย์ของมัน แต่หลังจากใช้งานไปแล้วพบว่ามีขีดจำกัดที่ยอมรับไม่ได้ในหลาย ๆ เรื่อง อาทิ

- ไม่สามารถแสดงข้อความครบ 80 คอลัมน์จริง ๆ ผู้สร้างระบบภาษาไทยบางระบบทำให้เกิดความรู้สึกแบบ 80 คอลัมน์ "เทียม" โดยการแทรก space เข้าไปหลาย ๆ ตัวแทน space ตัวเดียว เรียกกันว่าเป็นแบบ "ชดเชยสระ" การชดเชยบางครั้งก็ชดเชยกันทุก ๆ space บางครั้งก็ชดเชยเฉพาะเขตซึ่งมี space ติด ๆ กันเกิน n ตัว (n = 2 หรือ 3) ทำให้การจัดคอลัมน์ในตารางหรือรายงานดูเป็นระเบียบขึ้น
- ทำให้ซีพียูคำนวณได้ช้าลง ประมาณ 5-20% ทั้งนี้ขึ้นอยู่กับเทคนิคการออกแบบสูตรการคัดไทย (การช้าลงนี้บางครั้งวัดด้วย Norton SI หรือ SPEED ไม่ได้)
- บางครั้งเสียความรู้สึก โดยเฉพาะอย่างยิ่งการกดปุ่ม PgUp/PgDn ใน text editor หรือการ scroll ภาพ

■ ก. ภาพจากจอจริง



■ ข. ภาพจากจอเงา



รูปที่ 5 แสดงการทำงานของระบบจอเงาภายใต้ระบบภาษาไทยและเทลบิช (ใช้เทอร์โบปี 1.0 เอดิเตอร์) เทอร์โบปีเขียนข้อความภาษาไทยแบบไม่จัดระดับในจอจริง (ภาพ ก.) ผู้ใช้จะอ่านข้อความเป็นไทยตามปกติในจอเงา (ภาพ ข.) กรอบสี่เหลี่ยมคือไซด์คิกเมนเมนู ซึ่งมีผลต่อการจัดคอลัมน์ภาษาไทย

เร็วๆ บางบรรทัดอาจจะแหว่งขาดหายไปได้

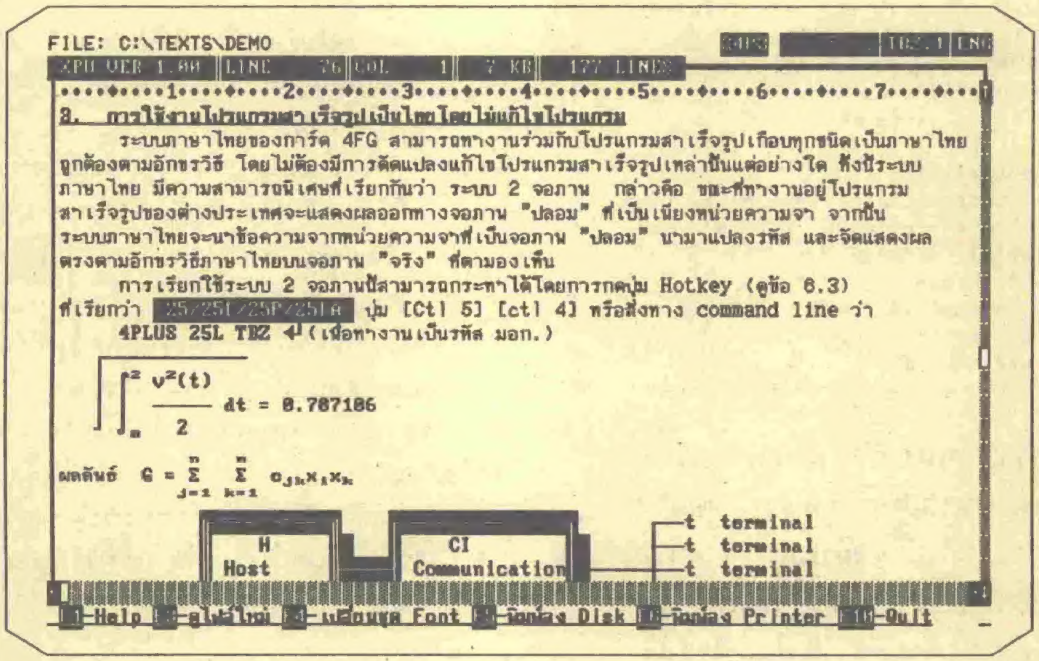
- โดยทั่วไปแล้วผลออกมาไม่สวย โดยเฉพาะอย่างยิ่งเมื่อนำมาเทียบกับโปรแกรมซึ่งแก้ไขเป็นไทยให้สมบูรณ์แบบ หรือผลิตขึ้นในประเทศไทย (ดูรูปที่ 6 เทียบกับรูปที่ 5)

อย่างไรก็ตาม Shadow screen processing มีความดีมากพอที่จะดึงดูดให้บรรดาผู้ผลิตไทยการ์ดทั้งหลายหันมาใช้กันเป็นจำนวนมาก โดยเฉพาะอย่างยิ่งนับตั้งแต่เดือนสิงหาคม 2530 เป็นต้นมา

วิดีโอฟังก์ชันอื่นๆ

นอกเหนือไปจากวิดีโอฟังก์ชันต่างๆตามที่ปรากฏในรูปที่ 7 (AH = 00 ถึง 1Ch) แล้ว การออกแบบโครงสร้างของระบบภาษาไทยจะต้องคำนึงถึงความครบถ้วนของมาตรฐาน ประสิทธิภาพของการทำงาน และการนำซอฟต์แวร์มาใช้อีก (software reusability) ผู้เขียนจึงเสนอให้ TSA มีการเพิ่มเติมวิดีโอฟังก์ชันเข้าไปบางส่วนเรียกหมายเลขของฟังก์ชันใหม่ว่า 45h แม้มีหมายเลขเดียวแต่ภายในจะถือว่ามีอีกหลายๆ ฟังก์ชัน ซึ่งพอที่จะแบ่งเป็นกลุ่มได้ดังนี้

1. Write String Extended (เป็นฟังก์ชันการเขียนข้อความที่เหนือกว่าฟังก์ชัน 13h ของไอบีเอ็ม ดังที่เสนอมาในบทความชุดนี้ ตอนที่ 1-7)
2. Write Multiple-string (ให้เขียนข้อความหลายๆ ข้อความสำหรับการแสดงเมนูที่มีประสิทธิภาพสูงสุด)
3. Write Box (ใช้เขียนกรอบสี่เหลี่ยมแบบต่างๆ)
4. Paint Attribute (ใช้ละเลงสีหรือแอตทริบิวต์ลงบนจอภาพ)
5. Control Software Switches (ใช้ควบคุมซอฟต์แวร์สวิตช์ต่างๆที่ใช้อยู่)
6. Push/Pop Screen (ใช้เก็บจอภาพที่ถูกซ้อนทับโดย pop-up menu)
7. Write to extended keyboard buffer (ใช้ใส่ข้อความลงในบัฟเฟอร์ของคีย์บอร์ดแทนการพิมพ์ด้วยมือจริงๆ)
8. Text Processing Routines (ใช้ประมวลผลข้อความภาษาไทย เช่น การวัดความกว้างของสตริง การสลับลำดับสระและพยัญชนะต้น เพื่อทำการ Sort การเปรียบเทียบการมาก่อน/หลังของคำไทย 2 คำ การหาจุดเริ่มและจุดจบของคำในสตริง ฯลฯ) รูทีน



รูปที่ 6 แสดงภาพภาษาไทยที่ควรจะเป็นจากข้อความในรูปที่ 5 โปรแกรมที่ผลิตในประเทศไทยล้วนจะสามารถแสดงข้อความได้เต็ม 80 คอลัมน์ และแสดงข้อความได้ถูกต้องสวยงาม ไม่ปนตัวอักษรไทยกับตัวอักษร Line graphic ภาพนี้บันทึกจากโปรแกรม XPU ท่านที่สนใจโปรแกรมนี้โปรดศึกษาจากบทความเรื่องนี้ในตอนที่แล้ว (โปรแกรม ViewFile)

AH	ความหมาย
00h	Set Mode
01h	Set Cursor Type
02h	Set Cursor Position
03h	Read Cursor Position
04h	Read Light Pen Position
05h	Select Active Display Page
06h	Scroll Active Page Up
07h	Scroll Active Page Down
08h	Read Attribute/Character at Current Cursor Position
09h	Write Attribute/Character at Current Cursor Position
0Ah	Write Character at Current Cursor Position
0Bh	Set Color Palette
0Ch	Write Dot
0Dh	Read Dot
0Eh	Write Teletype to Active Page
0Fh	Read Current Video State
10h	Set Palette Registers
11h	Character Generator
12h	Alternate Select
13h	Write String
14h	Load LCD Character Font/Set LCD High-Intensity Substitue
15h	Return Physical Display Parameters for Active Display
16h to 19h	Reserved
1Ah	Read/Write Display Combination Code
1Bh	Get Functionality/State Information
1Ch	Save/Restore Video State
45h	(ดูบทความชุดนี้ตอน 1-7) Write string Extended Write multiple-string Write box Paint attribute Write short data, long data, time to screen (extended TTY service) Control software switches Push screen/Pop screen Write to extended keyboard buffer Text processing routines and many more

รูปที่ 7 ตัวอย่างฟังก์ชันที่ใช้บริการเกี่ยวกับจอภาพ หมายเลข 00 ถึง 1C มีความหมายตามที่ไอบีเอ็มกำหนดให้ หมายเลข 45h เป็นของ Thai Systems Architecture เพิ่มเข้าไปเพียงฟังก์ชันเดียว แต่แยกออกเป็นหลายๆฟังก์ชันย่อยๆ ตามค่าของ AL ทั้งนี้เพื่อลดโอกาสเสี่ยงที่จะไปชนกับความหมายของหมายเลขอื่นๆที่ไอบีเอ็มอาจจะกำหนดขึ้นมาเพิ่มเติมภายหลัง

ส่วนนี้มักจะปฏิบัติกับตัวแปรในหน่วยความจำไม่แสดงผลออกทางจอภาพ

9. Extended Teletype Service (ใช้ควบคุมจอภาพโดยอาศัยคำสั่งที่ประสมอยู่กับข้อความซึ่งจะส่งออกทาง teletype (หรือ con: device) อาทิ การเปลี่ยนแอตทริบิวต์ การแสดงวันที่และเวลาขณะนั้นลงบนจอภาพ เป็นต้น)

จากที่ได้ทดลองทำไปแล้ว พบว่าการขยายวิดีโอฟังก์ชันออกไปตามแนวนี้มีประโยชน์กับการพัฒนาโปรแกรมใช้งานได้มากพอสมควร ประโยชน์ที่ได้รับมีดังต่อไปนี้

- การแสดงผลเป็นภาษาไทย/ตีกรอบ/overlapping window ด้วยความเร็วสูงทำได้ง่าย
- โปรแกรมใช้งานสั้นลง เพราะแทบจะไม่มีความรู้เกี่ยวกับอักขระวิธีเหลืออยู่ในโปรแกรมเลย
- การทำงานต่างๆ ค่อนข้างจะได้มาตรฐาน คือทุกโปรแกรมทำของง่ายถึงของยากได้ทัดเทียมกัน (เพราะรูทีนภาษาไทยมีอยู่ในหน่วยความจำ พร้อมที่จะใช้บริการตลอดเวลา)
- ช่วยในการดัดแปลงซอฟต์แวร์ต่างประเทศให้ทำงานเป็นไทยสมบูรณ์แบบได้ง่ายขึ้นมาก
- การบำรุงรักษาโปรแกรมง่ายมาก เพราะหากมีความผิดพลาดเกี่ยวกับภาษาไทยเกิดขึ้น แก้ไขที่ระบบภาษาไทยเพียงแห่งเดียวจะแก้ปัญหาให้กับโปรแกรมทุกโปรแกรม
- ฝึกให้โปรแกรมเมอร์ทำงานเป็นระบบมากขึ้น ทำให้เกิดความพร้อมที่จะศึกษาระบบอื่นๆที่อยู่ในรูปนี้เรียบร้อยแล้ว อาทิ Microsoft Windows 2.0, OS/2 Macintosh OS

อย่างไรก็ดีการออกแบบยูทิลิตี้รูทีนทั้ง 9 กลุ่มที่เสนอมานี้จะต้องออกแบบอย่างระมัดระวัง รูทีนเกือบทั้งหมดจะต้องเขียนแบบ re-entrant เพื่อให้ใช้งานจากหลายโปรแกรมได้พร้อมๆกัน โดยไม่ต้องรอให้จบการทำงานของรูทีนหนึ่งรูทีนใด สำหรับรายละเอียดปลีกย่อยของรูทีนเหล่านี้ยังไม่สามารถจะนำมากล่าวในบทความนี้ได้เพราะมีจำนวนมากเกินไป (ขณะนี้มีประมาณ 150 รูทีน)

โครงสร้างของระบบการพิมพ์

หัวใจของการพิมพ์ข้อความภาษาไทยในปัจจุบันก็คือการจัดบัฟเฟอร์ไว้ 3 แถว เพื่อใช้พิมพ์ข้อความ 1 บรรทัด เรียกว่าเป็น 3-pass (จะทำจริงๆ เพียง 2 แถวก็ได้ ไม่มีใครห้าม) ทุกๆ ตัวอักษรที่ถูกสั่งพิมพ์ (ดูรูปที่ 8) ก็จะถูก "จัดระดับ" เพื่อ

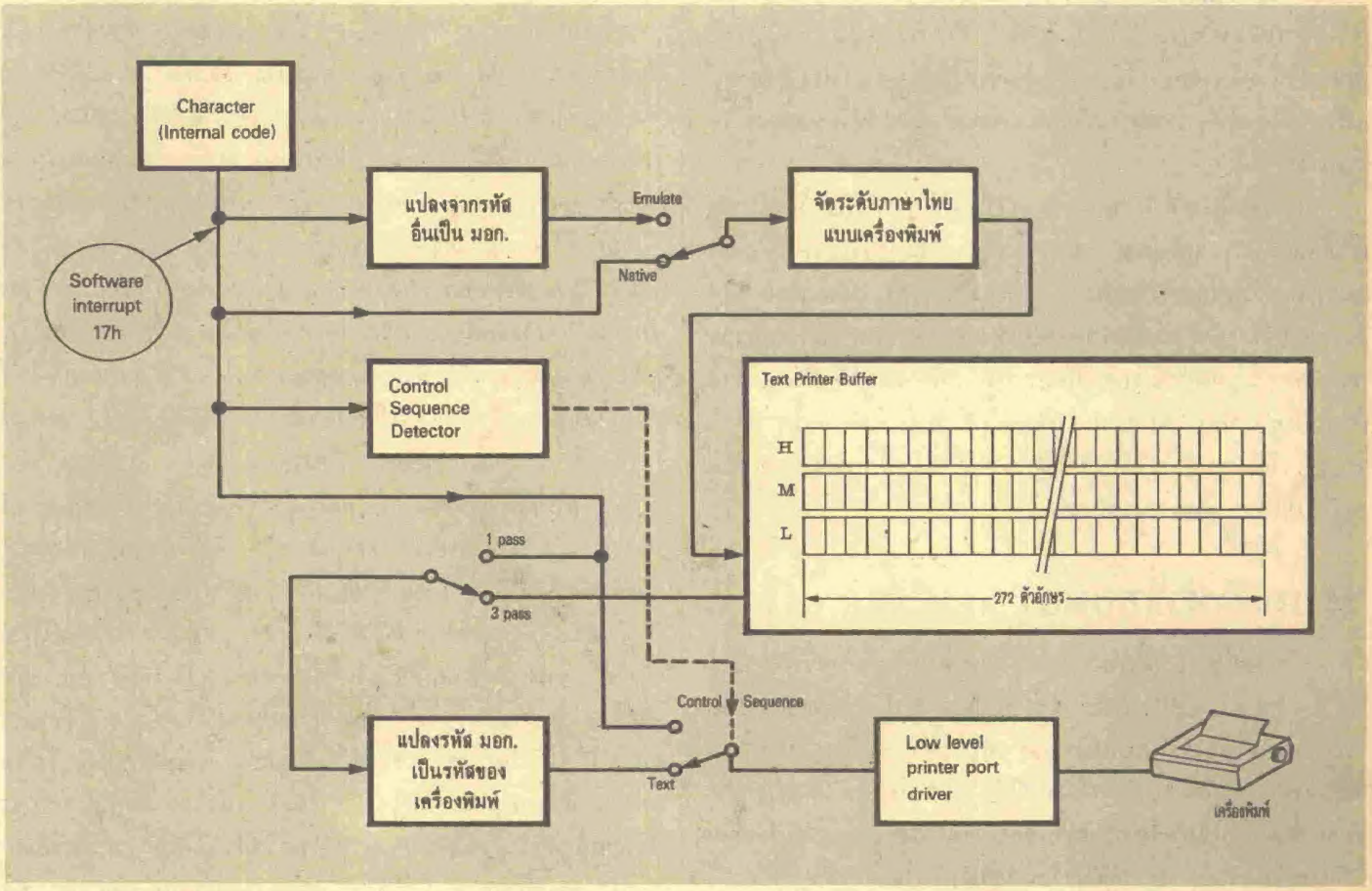
แยกว่าตัวอักษรตัวนี้ควรจะอยู่ระดับบน(H) กลาง(M) หรือล่าง(L) โดยถือว่าอักขระที่วางกลางบรรทัดเป็น M และทำให้ตำแหน่งการพิมพ์เคลื่อนไปทางขวา หากเป็นอักขระประเภทสระบนบรรทัดหรือใต้บรรทัด ก็จะถูกนำไปวางในบัฟเฟอร์ H หรือ L ตามความเหมาะสม การพิมพ์ออกทางเครื่องพิมพ์จริงๆแล้วเกิดขึ้นเมื่อได้รับตัวอักษร carriage return หรือ CR (แอสกี 0Dh)

การพิมพ์ตาม TSA ในรูปที่ 8 มีรายละเอียดเพิ่มเติมอีก 3 รายการคือ

(1) Lpt Ascii Table คือตารางใช้แปลงรหัสภายในที่ไม่ใช่ มอก.620-2529 ให้กลายเป็น มอก. ใช้เฉพาะกรณีที่กำลังทำงานด้วย Internal code ที่ไม่ใช่ มอก. (เรียกว่าเป็น emulated code)

(2) Printer Table คือ ตารางใช้แปลงรหัส มอก.ที่อยู่ในบัฟเฟอร์ออกสู่เครื่องพิมพ์ หมายความว่าเครื่องพิมพ์ที่นำมาต่อนั้นจะเป็นเครื่องพิมพ์รหัสใดๆก็ได้

(3) Control Sequence Detector คือรูทีนสำหรับดัก Esc หรือ FS sequence ที่โปรแกรมต้องการควบคุมเครื่องพิมพ์ คำสั่งพวกนี้อาจจะมี 1,2,3 หรือ 4 ไบต์ หากแปลงรหัสเป็นอย่างอื่นตามตาราง (1) และ (2) ข้างบนนี้ ก็จะทำให้ผิดความหมายได้ ในทำนองเดียวกันระหว่างช่วงที่เป็นการดาวน์โหลดตัวอักษร



รูปที่ 8 โครงสร้างของระบบการพิมพ์ข้อความภาษาไทย มองจากการไหลของข้อมูล สวิตช์ 1 pass/3 pass ใช้เลือกว่าจะพิมพ์แบบไม่จัดระดับหรือจัดระดับ สวิตช์ Emulate/Native ใช้เลือกว่ารหัสภายในเป็นรหัสอื่น (จึงต้อง Emulate) หรือรหัส มอก. (Native) ส่วนสวิตช์ Control Sequence/Text เป็นการควบคุมภายในโดยอัตโนมัติ ซึ่งระบบ Control Sequence Detector จะทราบเองว่าขณะนี้กำลังอยู่ระหว่าง escape sequence, FS-sequence, การดาวน์โหลดภาพตัวอักษร หรือระหว่างการพิมพ์ภาพแบบบิตแมป ซึ่งในช่วงนี้ห้ามแปลงรหัสและห้ามจัดระดับเด็ดขาด

หรือพิมพ์ภาพบิตแมปกราฟิกก็ห้ามแปลงรหัสเช่นเดียวกัน รูทีนประเภท Control Sequence Detector ควรทำงานควบคู่ไปกับ "ตารางคู่มือเครื่องพิมพ์" ซึ่งมีข้อมูลเกี่ยวกับ escape sequence ทุกๆ ตัวที่จะใช้กับเครื่องพิมพ์

สัญลักษณ์เครื่องพิมพ์ในรูปที่ 8 นั้น ในภาคปฏิบัติจะเป็น LPT1 หรือ LPT2 ก็ได้ ในกรณีที่มีบัฟเฟอร์ 3 แถวเพียงชุดเดียวนั้น การควบคุมให้เครื่องพิมพ์ 1 เครื่องพิมพ์ภาษาไทยได้ทั้งคู่นั้นก็ย่อมทำได้ แต่จะต้องสั่งพิมพ์ออกทางเครื่องพิมพ์เครื่องใดเครื่องหนึ่งให้ครบบรรทัด แล้วจึงจะย้ายไปพิมพ์ออกทางอีกเครื่องหนึ่งได้ ถ้าหากต้องการพิมพ์ออกสองเครื่องได้เป็นอิสระกันในระดับตัวอักษรต่ออักษรอาจมีความจำเป็นต้องสร้างบัฟเฟอร์ 3 แถวไว้เกิน 1 ชุด ลักษณะแบบหลังนี้เท่าที่ผู้เขียนใช้งานมาพบว่าไม่ใคร่จะเกิดขึ้น ปัญหาส่วนใหญ่สามารถแก้ไขได้ในระดับการจองเครื่องพิมพ์กันทีละบรรทัดเพราะเข้าใจง่ายกว่า

ตารางอื่นๆที่มีผลต่อการทำงานของเครื่องพิมพ์ที่เด่นชัดก็ได้แก่ ตารางเก็บขนาดช่องว่างระหว่างการพิมพ์ 3 pass ในรูปที่ 8 ผู้เขียนได้แสดงไว้ 3 ค่า คือ HM, ML และ LH จำง่ายๆว่า HM คือระยะการป้อนกระดาษระหว่างการพิมพ์สระ/วรรณยุกต์เหนือบรรทัดแถว(H) ค่า ML คือระยะการป้อนระหว่างแถว M กับ L (สระใต้บรรทัด) ฯลฯ เครื่องพิมพ์แบบ 9 เข็ม จะป้อนได้ละเอียด 1/216 นิ้ว ส่วนแบบ 24 เข็มจะป้อนได้ละเอียด 1/180 นิ้ว

ระบบภาษาไทยต้อง RE-ENTRANT

ในโลกของ MS-DOS มีปัญหาโลกแตกอยู่เรื่องหนึ่งก็คือว่า ดอสเป็นโปรแกรม ถ้าระหว่างการทำงานภายในดอสหากงานยังไม่จบ เกิดมีอินเตอร์รัพต์บางอย่างขึ้นมา โปรแกรมที่อินเตอร์รัพต์จะไม่มีสิทธิ์เรียกใช้ดอสซ้อนอีกครั้ง เช่น เพื่ออ่านไฟล์หรือเขียนไฟล์) หากเรียกใช้งานซ้อนจะรับประกันได้ว่าเครื่องจะแขวน (hang) แน่นอน อาการแบบนี้ปรมาจารย์ในต่างประเทศท่านว่า ดอสเป็นโปรแกรมที่ไม่ re-entrant คือระหว่างงานไม่จบจะเข้าซ้อนไปอีกครั้งไม่ได้ โปรแกรมประเภทเรซิเดนต์ต่างๆ เช่น ไซด์คิก ต่างก็ต้องใส่ระบบป้องกันดอสพังกันทั้งนั้น ทั้งนี้เพื่อรับประกันว่ากดเรียกใช้งานได้ทุกขณะ และเมื่ออยู่ในโปรแกรมประเภทนี้แล้วต้องรับรองว่าสามารถใช้ดอสอ่าน/เขียนไฟล์ได้โดยปลอดภัย เทคนิคที่ใช้กันก็คือต้องตรวจสอบว่าจบงานดอส หรือดอสปลอดภัยแล้วจึงจะยึดเครื่อง (การยึดเครื่องนั้นบางทีก็อยู่กลางการทำงานของดอส แต่เป็นฟังก์ชันที่ปลอดภัย)

โชคดีที่ ROM BIOS มีคุณสมบัติด้าน re-entrant ดีกว่าของดอส ดังนั้นความคิดที่จะมีการอินเตอร์รัพต์โปรแกรมขณะทำงานภายใต้ไบออส และเรียกใช้งานไบออสซ้อนเข้าไปอีกจึงเป็นได้ค่อนข้างง่าย (แต่เข้าใจว่าบางส่วนของไบออสก็ไม่ก็ไม่ re-entrant ตัวอย่างเช่น ดิสค์อินเตอร์รัพต์ หมายเลข 13h เป็นต้น)

การสร้างให้ระบบภาษาไทยทำงานแบบ re-entrant ได้นั้น ผู้เขียนพบว่ามีประโยชน์มาก และถือว่าเป็นความต้องการขั้นต่ำของการผลิตซอฟต์แวร์ระบบที่ได้มาตรฐานสากล ลองนึกภาพการทำงานของระบบไมโครคอมพิวเตอร์สมัยใหม่ดูซีครับ (เรื่องจริงไม่อิงนิยาย) *เราบูตเครื่องด้วย DOS 3.10 ใส่โปรแกรมระบบภาษาไทย 6FGPLUS (resident) ตามด้วยระบบสื่อสาร FK 2.00 (resident) ตามด้วยโปรแกรม Power 1.00 ซึ่งเป็นยูทิลิตี้สำหรับโปรแกรมเมอร์ (resident) ตามด้วย PEN text editor ซึ่งเป็นโปรแกรมธรรมดา ระหว่างการทำงาน โปรแกรม PEN วาดกรอบ วาดแอตทริบิวต์ และเขียนสตริงบนจอภาพโดยเรียกใช้ระบบภาษาไทยโดยไม่ใช้บริการของดอส ยกเว้นการอ่าน/เขียนลงไฟล์ ขณะทำงานอยู่ระบบสื่อสาร FK 2.00 ซึ่งต่อกับโมเด็มและสายโทรศัพท์อาจจะได้สัญญาณกริ่งโทรศัพท์ดังขึ้น ระบบ FK ก็จะเช็คดูว่า DOS ปลอดภัยหรือยัง (ถ้าอยู่ระหว่างการทำงานเกี่ยวกับไฟล์ก็จะมีการรอจนงานจบ) หากปลอดภัยก็จะเข้ายึดเครื่องและจอภาพทันที (โดยจะ push screen เก็บภาพเดิมบนจอภาพเข้าสู่ screen stack) จากนั้นก็จะแสดง alert box แจ้งให้ผู้ใช้ทราบว่า มีโทรศัพท์เข้ามาและจะยกหูโทรศัพท์ภายใน 3 ครั้ง แสดงผลการรับข้อมูล... ฯลฯ จนกระทั่งรับส่งข้อมูลเสร็จแล้วจึงจะ pop screen เดิมของ PEN ให้ผู้ใช้ทำงานต่อไปราวกับว่าไม่เกิดอะไรขึ้น การอินเทอร์รัพต์นี้เกิดขึ้นได้ทุกขณะ แม้กระทั่งในช่วงที่ PEN กำลังเขียนข้อความบรรทัดที่ 5 กลางๆบรรทัด (โดยใช้ระบบภาษาไทยฟังก์ชัน VXYAText) และยังเขียนไม่จบบรรทัด ส่วนโปรแกรม FK เองเมื่อยึดเครื่องไปได้แล้ว ระหว่างการทำงานของ FK (จะนานกี่ชั่วโมงก็ได้) จะมีการใช้ฟังก์ชันต่างๆ ของระบบภาษาไทยได้อย่างเต็มที่ และเมื่อออกจาก FK โปรแกรม PEN ก็จะกลับมาเขียนข้อความบรรทัดที่ 5 ต่อไปได้อย่างถูกต้อง เหมือนไม่เกิดอะไรขึ้น!!*

ระบบเดียวกันที่ว่านี้อนุญาตให้ผู้ใช้กดเรียกไซด์คิกขณะใช้งาน PEN หรือโปรแกรมอื่นๆ และเมื่ออยู่ในไซด์คิกอาจจะกดเรียก Power ซ้อนขึ้นมา และขณะอยู่กลาง Power ก็สามารถรับ/ส่งแฟ้มข้อมูลทางโทรศัพท์ได้ถูกต้องสมบูรณ์โดยที่จอภาพและงานของแต่ละโปรแกรมยังคงกลับมาทำงานต่อได้อย่างถูกต้องสมบูรณ์ ถ้าไม่ออกแบบเป็น re-entrant สิ่งเหล่านี้ก็จะเกิดขึ้นไม่ได้

จะใช้ Global Switch หรือ Local Switch?

เหตุการณ์ที่สลับซับซ้อนกว่าตัวอย่างข้างบนนี้อาจจะเกิดขึ้นกับ MS-DOS ในอนาคตอันใกล้ และเกิดขึ้นแน่นอนกับ OS/2 ซึ่งเราอาจจะมี resident program ใช้งานหลายอัน และอาจต่อเป็นเทอร์มินัลกับมินิหรือเมนเฟรม และอาจจะเป็นสมาชิกของ LAN ด้วยบนเครื่องเดียวกัน ลองนึกภาพดูว่าจะให้แต่ละโปรแกรมแยกกันทำงานเป็นเอกเทศหรือจะใช้ความรู้ภาษาไทยร่วมกันที่ระบบภาษาไทย? ระบบภาษาไทยที่ใช้ร่วมกันได้นั้นก็คงจะต้องปฏิบัติงานแบบ re-entrant และมีความสามารถที่จะแยกแยะได้ว่า สภาพแวดล้อมของภาษาไทย (คือสวิตช์ต่างๆที่พูดถึงในบทความนี้) จะเป็นสวิตช์แบบชุดเดียว (global) หรือแบบหลายชุด ต่างคนต่างมีในแต่ละแอพพลิเคชั่น (คือเป็น local switch)? เมื่อเราเปลี่ยน session ไปมาจากการทำงานกับไมโคร ไปสู่การโต้ตอบกับโปรแกรมบางประเภทกับมินิหรือเมนเฟรม สวิตช์ก็อาจจะต่างกัน อย่างไรก็ดีระบบภาษาไทยจะต้องบริหารสวิตช์ต่างๆเหล่านี้ได้อย่างมีประสิทธิภาพ อัตโนมัติ และสมเหตุสมผลพอสมควร สถานการณ์ที่ยุ่งที่สุดอาจจะเกิดขึ้นกับเครื่องที่อยู่ในสภาพคล้ายๆแบบนี้ก็ได้ เช่น (เรื่องที่ยังไม่เกิดขึ้น) *ใช้ PC ทำงานกับไทยการ์ด รหัส มอก. ขอใช้ Lotus 1-2-3 รหัสเกษตร และในเวลาเดียวกันขอค่า on-line กับมินิคอมพิวเตอร์ โดยใช้เทอร์มินัลเอมิวเลเตอร์ โดยกำหนดให้รหัสสื่อสารเป็น EBCDIC ทั้งหมดมี hot key สำหรับย้ายจอภาพไปเป็นของงานต่างๆ โดยไม่กระทบกระเทือนงานอื่นได้ตลอดเวลา*

มาถึงจุดนี้แล้วเราคงจะต้องคิดกันอีกว่า Table บางอันที่เสนอไว้ใน TSA นั้น ควรจะเป็นของส่วนกลาง (ในระบบภาษาไทย) หรือแยกไว้ในแอพพลิเคชั่น หรือจะจัดไว้ในระบบภาษาไทยโดยมีตารางละ 3-4 แบบ เพื่อการทำงานหลายๆรหัส โดยที่ในแต่ละแอพพลิเคชั่นจะจดจำตำแหน่งสวิตช์ของตนเองเอาไว้เป็นอิสระต่อกัน

ท่านที่มีประสบการณ์เกี่ยวกับ DOS environment variables คงจะซาบซึ้งดีว่า ในสภาพของการทำงานแบบมีหลายๆงานพร้อมๆกันนั้น โครงสร้างที่เรียกว่า environment นั้นเป็นสิ่งจำเป็น ในระบบภาษาไทยก็เช่นเดียวกัน การตั้งสวิตช์ต่างๆสำหรับแต่ละโปรแกรมอาจจะต้องแยกกันโดยเด็ดขาดคล้ายๆกับ environment ของดอส

ทรัพยากรที่ไม่สามารถใช้แบบ RE-ENTRANT

การออกแบบให้ระบบภาษาไทยมีฐทีนประเภท re-entrant ทั้งหมดนั้นเป็นสิ่งที่ดี แต่มีบางรายการที่ผู้ใช้อาจจะต้องคำนึงถึงว่าเป็นไปไม่ได้ในชีวิตจริง ยกตัวอย่างเช่น การทำ Print Screen ลงกระดาษ ไม่สมควรจะทำงานแบบ re-entrant เพราะผิดธรรมชาติ คงไม่มีผู้ใดต้องการพิมพ์จอภาพไปได้ครึ่งทาง เพื่อที่จะหยุดพิมพ์ภาพอีกหนึ่งภาพที่สมบูรณ์ แล้วจึงจะกลับมาพิมพ์ภาพที่เหลือต่อให้จบ การพิมพ์รายงานหรือพิมพ์ไฟล์ออกกระดาษก็เช่นเดียวกัน จะต้องไม่ re-entrant งานพิมพ์งานหนึ่งจะต้องจบก่อนจะพิมพ์งานใหม่ออกมา การจบนั้นหากจำเป็นก็อาจจะจบแบบถูกบังคับก็ได้ ลักษณะของเครื่องพิมพ์ที่กล่าวมานี้เป็นตัวอย่างของ "ทรัพยากรที่ไม่สามารถใช้แบบ RE-ENTRANT" (หรือ exclusive resource)

อุปกรณ์อื่นๆที่จะต้องระมัดระวังคล้ายๆกับเครื่องพิมพ์ก็เห็นจะได้แก่ คีย์บอร์ด โมเด็มและจอภาพ อุปกรณ์เหล่านี้จะใช้แบบ exclusive หรือใช้ร่วมกันก็ได้ แต่จะผิดธรรมชาติหรือไม่นั้นต้องว่ากันโดยรายละเอียด ตัวอย่างปัญหาที่น่าคิด 2-3 ข้อที่พอจะเล่าสู่กันฟังได้ดังต่อไปนี้

1. ปัญหาคีย์บอร์ด ขณะใช้กับโลตัสหรือดีเบส เรากำลังจะป้อนภาษาไทย แต่บังเอิญไปกดปุ่มใช้งานไซด์คิก ซึ่งลงเอยต้องการใช้แป้นพิมพ์เป็นอังกฤษ กลับสู่โลตัสควรให้แป้นพิมพ์คงความเป็นอังกฤษหรือไม่?
   คำตอบ [ ] ควรแยกสถานะระหว่างไซด์คิกกับภายนอก
   [ ] ควรเป็นสถานะเดียวตลอดเวลาเช่น NumLock, ScrollLock ที่ใช้กันอยู่
2. ปัญหาโมเด็ม ขณะกำลังรับส่งไฟล์และถูกโปรแกรมอื่นอินเตอร์รัพต์ไป โปรแกรมใหม่นี้จะขอใช้ COM: port อันเดียวกับโมเด็ม จะป้องกันหรือจะยอมให้เกิดขึ้นได้ไหม? ทำไม?
3. ปัญหาจอภาพ หากระบบเขียนจอภาพเป็น re-entrant จริง เราจะป้องกันจอเลอะจอเสีย เพราะมีข้อความจากโปรแกรมที่มาอินเทอร์รัพต์งาน (แต่ทิ้งข้อความไว้บนจอ) ได้อย่างไร?

สรุป

การเดินทางสู่ Thai Systems Architecture สำหรับ OS/2 และสำหรับเทอร์มินัลอีมูเลเตอร์เพื่อต่อกับมินิ/เมนเฟรม ยังมีระยะทางอีกไกลจึงจะถึงจุดหมายปลายทาง ผู้เขียนหวังว่าบรรดาผู้อ่านที่เริ่มสนใจการสร้างระบบภาษาไทยและผู้สร้างระบบภาษาไทยทั่ว ๆ ไปคงจะได้ประโยชน์บ้าง จากแนวทางทางการออกแบบฯนี้ หากท่านมีข้อคิดเห็นหรือพบข้อผิดพลาดใดๆในบทความ กรุณาแจ้งให้ผู้เขียนทราบด้วยจะเป็นพระคุณอย่างยิ่ง